卒業論文

緩やかに発進停止できる全方向移動型歩行訓練機

高知工科大学
知能機械システム工学科
知能ロボティクス研究室
1040125
近澤 敦史

第 1 章 序論

1-1. 研究背景

現代社会の課題として元気のない高齢者の数が増加している。人が加齢するにつれ、歩行の量が次第に減り、病気に掛かりやすくなる。歩行という運動は日常の生活の一部ではあり、有酸素運動である。ダイエット効果や、ストレス解消、また知的作業をする際に歩行をすると考えが纏まるなどの効用があると認めている。そこで高齢者の健康状態を保つため、歩行運動が必要である。もし、何らかの原因で歩行障害が起こると、歩行訓練を行うことが必要である。ここで、歩行訓練のリハビリを考えてみることにする。リハビリは、患者一人ではできない場合は、必然的に理学療法士や、ヘルパーなどの助けが重要になる。しかし、保険制度による規則などが原因で患者一人あたりにつける理学療法士の時間が十分に確保されているとは言いがたい。また、理学療法士の身体的負担なども計り知れない部分がある。そこで、理学療法士やヘルパーなどの負担を減らすため、研究室では転倒防止機能を有した、全方向移動機能を持つ歩行支援器を試作した。しかし、現状ではスピードが速く、また発進時･停止時の振動があり、リハビリに適しているとはいえない。そこで、スピードの調節をし、発進時･停止時の挙動を緩やかにした。緩やかに発進停止できる全方向移動型歩行訓練器の開発をした.

1-2. 研究意義

「全方向移動型歩行訓練機」作成の意義として、

1. 自立してリハビリが行える
2. 介護者の負担が減少する
3. 患者が介護者に対する不安が減少する
4. 全方向行きたいように歩行ができる

以上の通りである。これらは非常に肝心な要項である。
理由は以下に述べる。

1. 自立して歩行が行えるという点は患者自らが自分の意思で歩行訓練を行えるということである。

2. 次に、介護者の負担が減少するだが、従来の介護では介護者は訓練中ずっと患者に付きっきりだがこの「全方向移動型歩行訓練機」では離れた場所から患者の歩行の仕方が観察できる。

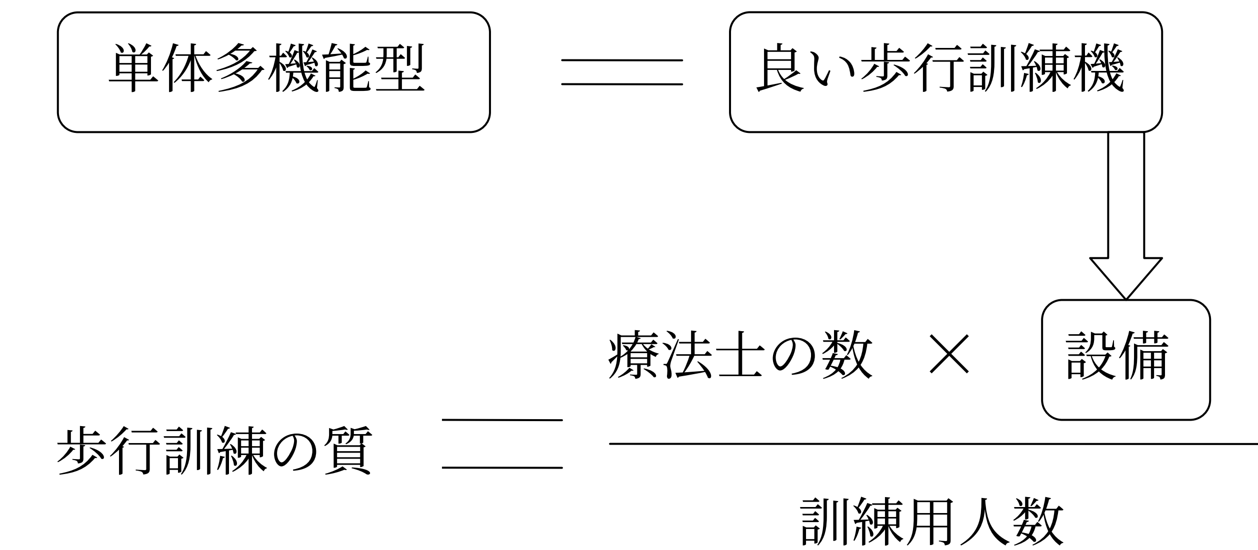
3. 患者の方の心理としては、自分が歩行できないことで、介護者の方に負担が掛かっているのではないかという自責の念に駆られているが、「全方向移動型歩行訓練機」では介護者は離れたところから観察しているので、患者の心理的負担も軽くて済む。

4. 「全方向移動機構型歩行器」では、患者の行きたい方向に行くことが可能なので、従来のリハビリ訓練では不可能だった、横方向の移動や、斜め方向の移動などができ、実生活においても利用のできる歩行が可能である。

このような理由で「全方向移動型歩行訓練機」の有用性を知ることができる。

1-3. 歩行訓練の質とは

式（1. 1）は歩行訓練の質の概念的な式である。



式（1.1）

式(1.1)の意味は、質の高い単体機能型の歩行訓練機を導入することにより歩行訓練の質を上げようとするものである。

それには転倒防止でき、且つ全方向移動可能な機構を有した、歩行訓練機の開発が望れている.

1-4.　全方向移動機構を有した歩行器の研究

関連研究

① 天井吊り下げ式「フローラ」

② SRC ウォーカー(Spontaneous Reaction Control Walker)

以上の二つが挙げられる。その特徴を以下に示す。

① 天井吊り下げ式「フローラ」　図(1.1)参照

フローラの３大特性

① 天井を自由自在に移動できる
② 重力(体重)を軽減または無力化できる
③ 長時間装着しても痛くならない



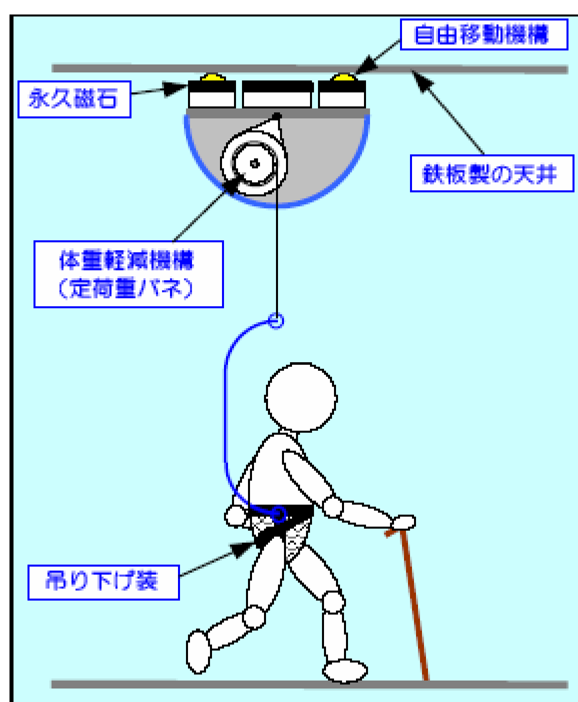
図(1．1)

「歩行支援システム(フローラ)」図(1.1)は、図(1.2)に示すように鉄板製の天井を永久磁石で吸着し、そこから人を吊り下げている歩行支援システムであり、自由自在に移動することが可能。

またフローラは、体重軽減力を変えることができるので、体への負担が小さく、歩行に障害のある方の意思で歩くことができる。

フローラ/仕様

・走行速度: 0〜0. 7 m/s (調整式)
・走行制御: 8 方向制御(ジョイスティック時)
前進・後進、前進の左・右曲がり、後進の左・右曲がり、左・右のその場旋回
3 方向制御(杖等への組み込み時)
前進、左・右のその場旋回
・吊り上げ力: 定荷重ばね(調節式)
・軽タイプ: 15 〜 30 kg
・中タイプ: 30 〜 50 kg
・銃タイプ: 50 〜 70 kg



図(1. 2)

特長
①自由歩行・健常者と空間を共有
②転倒防止機能・体重軽減機能
③日常的に使用可能
④自立支援・自発リハビリ支援

以下に詳細を述べる。

① 従来のレール方式では移動範囲が限られていたが、フローラはその機構の使用できる範囲内ならばどこでも自由に歩行可能

② 定荷重ばねで体を吊って体重を大きく軽減させているので、転倒しにくく、怪我をしにくい

③ 任意の姿勢をとることができ、長時間装着できる装具であり、視野が高いので自分の歩行の分析できる

④ 従来の移動リフトなどでは、完全に吊り上げるだけで、人を物として扱っているような感じがし敬遠されがちだったが、フローラでは自分の意思で無理なく自発的に歩行することが可能

使用対象

① 歩行訓練機装置
② 転倒防止装置
③ 子供用の歩行訓練装置
④ 移動介助装置(移乗用リフトの代替)
⑤ 日常生活歩行支援装置
⑥ 老人向け健康増進クラブ
⑦ 職業訓練施設
⑧ 介助省力化装置

説明:

① 脳卒中(脳血管障害)による片麻痺患者、骨折患者、脊髄損傷患者などの歩行訓練用

② フローラを転倒防止装置として使用することにより、介助者が不要、または、介助の負担が減少

③ 骨折・障害などの子供の歩行訓練、脳性麻痺の子供の正常発達を促すための歩行獲得訓練、この機構を用いることにより、リハビリを ” 遊び感覚 ” で楽しみながらの歩行訓練が可能

④ 移乗リフトによる介護に頼ることなく、自助自立した生活を送り、積極性を維持して、精神的にも快活にすごすことが可能

⑤ 老人ホームなどで、足腰が弱り車椅子に頼りがちな老人の歩行訓練・歩行支援用

⑥ 足腰の強さを維持向上させるための「介護予防用」

⑦ 車椅子が前提となっている職リハ段階などで、実施可能な作業の範囲が大幅拡大

⑧ 患者を抱き起こすなどの介助動作は重労働で腰痛などの原因のひとつだが、フローラにより、介助力がない方でも介助可能

機構

① 鉄板製の天井
② 自由移動機構
③ 体重軽減機構(定加重ばね)
④ ジョイスティック
⑤ 吊り下げ装具

説明:

① 従来のバリアフリーの考え方と根本的に異なり、床の整備なしで、天井に鉄板を装備するだけで健常者と被介助者が空間を共有

② 鉄板製の天井に強力な永久磁石で安定して吸着しており、自由自在に移動歩行することが可能で、定加重ばねの設定値以上の力は作用しないので、装置は絶対に落下しない

③ 定加重ばねの設定加重値を可変させることにより、個人の体重や状態に合わせた介助力を得ることが可能

④ 移動歩行操作はジョイスティックで制御、ジョイスティックの他に、歩行補助杖(ロフストランドクラッチ等)に組み込むことも可能で、操作が簡単

⑤ 従来のスリングと異なり、すべての姿勢を考えた画期的な装着具で痛くなるようなところはない

・課題点

・左右への移動ができず、全方向移動はできない

・フローラの機構が巨大なために使用スペースが限られ、施設があるところでしか使えない

SRC ウォーカー(Spontaneous Reaction Control Walker)



図(1. 3)

SRC ウォーカーの写真を図(1. 3)に示す。

この歩行器は重度の障害児のために開発された。重度の障害とは脳性麻痺をはじめとする運動発達障害児に対する歩行訓練を目的としている。

これまでも様々なタイプの歩行器が障害児のために利用されてきた、いずれの歩行器も、グリップ(握り)、もしくはU字型のアームレスト(前腕支持具)が付属されていた。基本的にグリップを持つか、アームレストで前腕を支持して移動を補助する、Walking Aids である. 従来からある歩行器は両下肢で体重が支持でき、上肢で全身のバランスをとることが可能なケースのみ使用することができた。しかし、座位保持が困難な重度児にも歩行器を適応することは特殊な例を除きなかった。

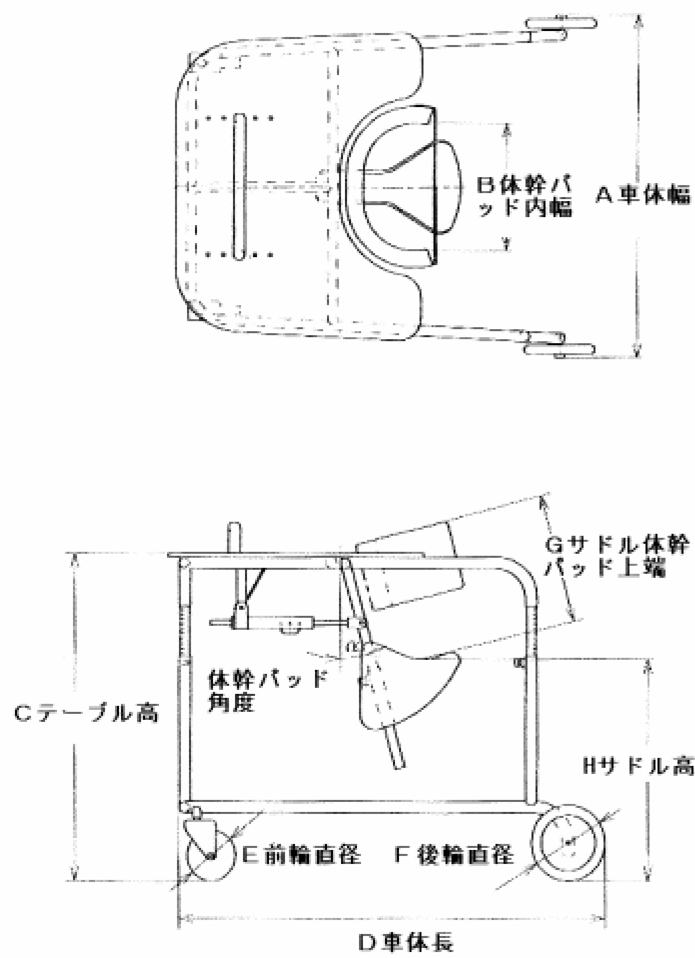
SRC ウォーカーの特徴

SRC ウォーカーは重度障害児を対象に作製された、移動補助具なので以下に示す点が一般の歩行器と違う

① 体幹を前傾させて姿勢を保持する
② サドルにより体重を支え、両足の交差を防ぐことが可能
③ テーブルによって上肢による支持を促す
④ 両下肢で後方に蹴り出して前進することが可能
⑤ 前に進もうとする構えと意識付けができる
⑥ 成長や運動発達に合わせ、各部の寸法と角度の調整が可能 参照 表(1.1)

記号	項目	サイズ／寸法				単位
		SS	S	L	LL	
A	車体幅	530	675	740	765	mm
B	体幹パッド内幅	190	240	270	270	mm
C	テーブル高	450〜600	500〜700	650〜900	760〜1100	mm
D	車体長	750	770	870	930	mm
E	前輪直径	75	75	100	100	mm
F	後輪直径	130	150	150	150	mm
G	サドル〜体幹パッド上端	280〜360	290〜420	300〜470	300〜470	mm
H	サドル高	160〜380	180〜500	250〜760	370〜960	mm
α	体幹パッド前傾角度	0〜45	0〜45	0〜45	0〜45	度

表(1.1)



図(1. 4)

上に SRC ウォーカーの平面図と側面図を示しておく。

課題点

・障害者の成長に合わせて、購入していたのではコストが掛かりすぎる。

第2章　全方向移動型歩行訓練機

全方向移動型歩行訓練機の定義

本研究の題材、全方向移動型歩行機の定義をしておく。

(1) 歩行器の回り(360 度)全ての方向に対し、直進が可能
(2) その場で 360 度回転することで、全ての方向に歩行器の正面を向けられる

2－1. 全方向移動型歩行訓練機の必要性

ここで全方向移動型歩行訓練機の必要性を示す。現在病院や、施設で行われている、前後移動のみの歩行訓練だけでは不完全に感じる。理由として、患者が外出すると、今までのような平坦な訓練場とは路面の状況が異なり、患者の方以外の人間も当然道路を通行している。また歩道の段差や、階段の昇降などもしなければならない。最近はバリアフリーの施設が増えてきてはいるが、バリアフリー整備もすべての場所にまで及んでないのが現状である。外界からの思わぬ刺激により、体のバランスを失い体勢が崩れる、あるいは人との衝突を避けるため斜め方向や、真横に動き衝突を避けなければならないという事態が予測される。その時の咄嗟の行動は、訓練をしていなければ体を動かすことができないと考えられる。通常、健常者の歩行は、左右前後の単純な歩行に加え、斜めへの移動や、その他複雑な動きの組み合わせであり、少なくとも歩行訓練時の訓練移動範囲を広げるだけで、より健常者に近い歩行が可能であると考える。その意味で、全方向に移動可能な歩行訓練機は大変意義のあるものだと考えられる。
以上の理由より｢全方向移動型歩訓練器｣の重要性をうかがい知ることが可能である。

2－2. 全方向移動型歩行訓練機

図(2.1)が開発した｢全方向移動型歩行訓練機｣の写真である。
この歩行訓練器の特長は、オムニホイルを使用することによって、前後･左右の移動はもちろん、斜め移動もできる。また操作方法もジョイスティックを用いて行い、緊急の際にはいつでも停止できるよう、胴回りに緊急停止ボタンも備え付けてある。
以下にそれぞれの役割を示す。



図(2.1)

2-2-1. オムニホイル

「全方向移動型歩行訓練機」に用いたオムニホイルの写真(2.2)を掲載しておく。

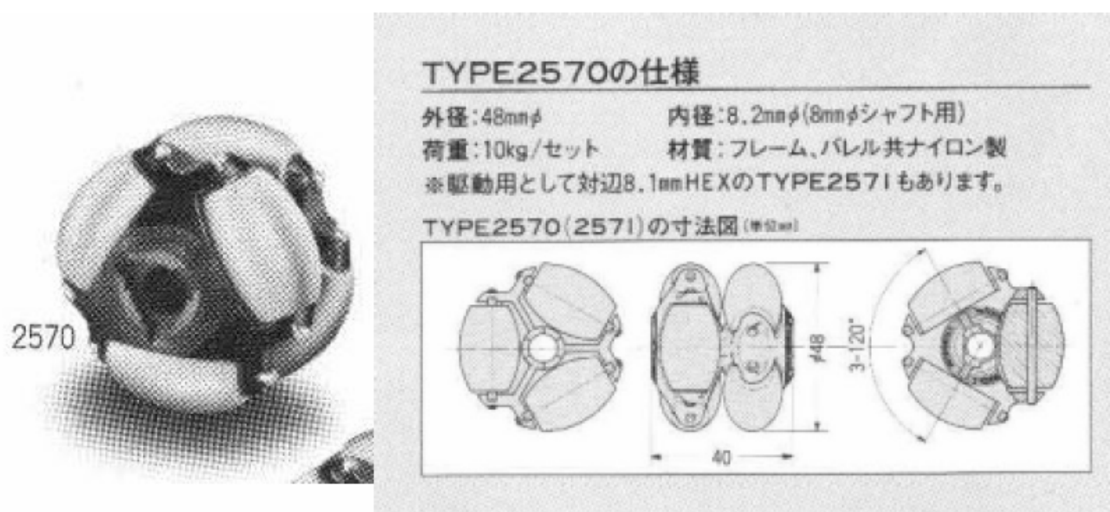
このオムニホイルは x 軸周りの回転･y 軸周りの回転･z 軸周りの回転が可能でそれにより前後の移動はもちろん、従来の歩行支援器では不可能だった真横への移動や斜めへの移動も可能になった。



図(2.2)

次ページの図(2.3)が、(株)富士製作所の作成した、オムニホイルの仕様である。

オムニホイルのとは、車軸に垂直方向の軸を車輪の周囲に設け、その軸にローラを取り付けた車輪であり、このことより走行面をホロノミックに移動することが可能である。



図(2.3)

2-2-2. ジョイスティック



図(2.4)

図(2.4)に示した、ジョイスティックを今回使用した。
このジョイスティックは justy の JSK - 104 という型番である。
全方向移動型歩行訓練機にはこのジョイスティックを 2 本使用した。以下に、操作方法を示す。

操作方法

1.両スティック:前倒し→前進
2.両スティック:後倒し→後進
3.両スティック:右倒し→右方向移動
4.両スティック:左倒し→左方向移動
5.両スティック:右斜め前方向→右斜め前方向移動
6.両スティック:左斜め前方向→左斜め前方向移動
7.両スティック:右斜め後方向→右斜め後方向移動
8.両スティック:左斜め後方向→左斜め後方向移動

9.右スティック:前方向･左スティック:後方向→左回転
10.右スティック:後方向･左スティック:前方向→右回転

機能として、ｘ-ｙ 軸調整の調整ができる。

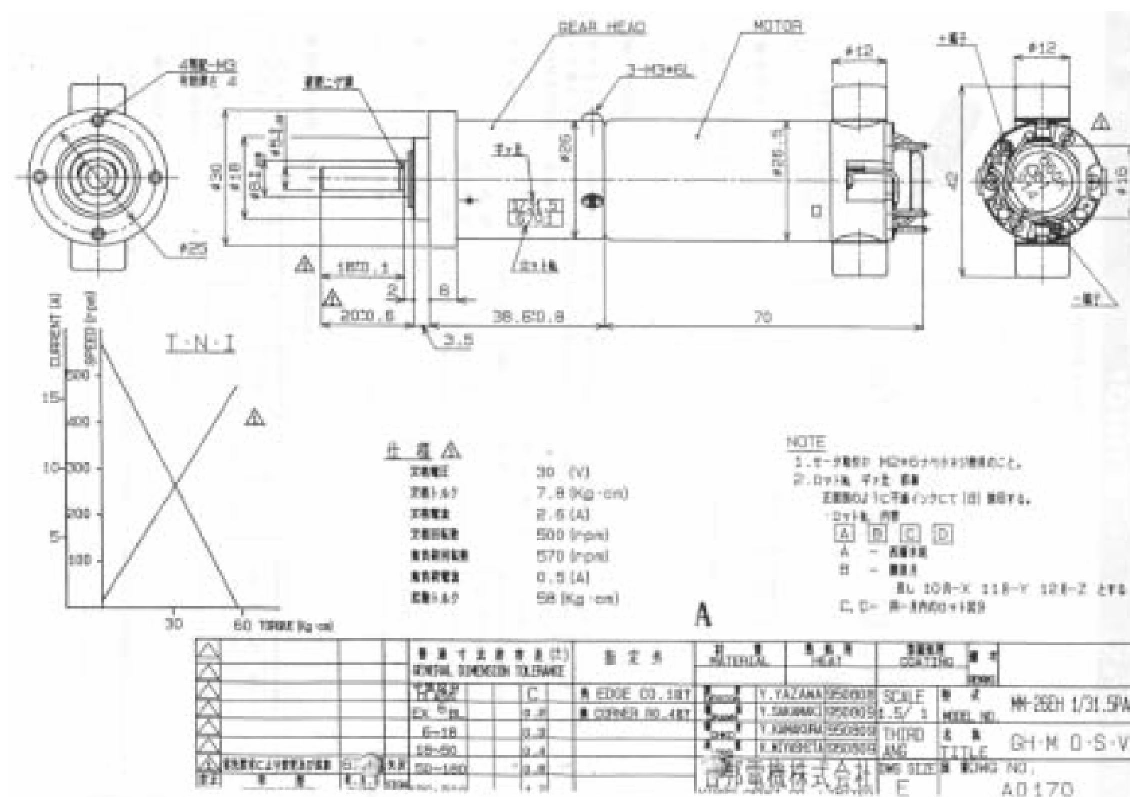
2-2-3. モータ



図(2.5)

図(2.5)に示したモータが今回使ったモータである。
このモータは、日邦電気株式会社で作成された、型式 MM-26EH で全方向移型歩行支援器には各車輪に一つ使用し、計 4 つ使用している。

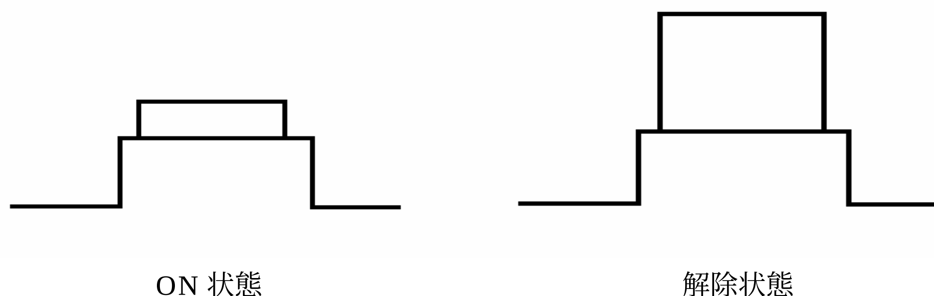
図(2.6)に、 MM-26 の仕様を示した。この MM-26 の定格トルクは定格出力 30(W)で 7.8kg/cm という大出力のモータである。また、このモータの定格出力は 2.6(A)と消費電圧も少なく、大変使い勝手がよい。



図(2.6)

2-2-4. 緊急停止ボタン

次ページに緊急停止ボタン図(2.7)を掲載した。また、この紹介したものもある。



ON 状態　　　　解除状態

緊急停止ボタンが ON 状態ならばモータは回転せずに停止しており、解除状態のときはモータが回転し、オムニホイルが起動する。

次ページに操作マニュアルを記載しておく。一部 2-2. 全方向移動型歩行訓練機と重複する項目もある。



図(2.7)

この緊急停止ボタンの説明は次に示す。

2-3. 操作マニュアル

1. 起動方法

① Key をまわす

② 緊急停止ボタンを解除する

2. 操作方法

2-1. ジョイスティックを操作する

詳細:

１. 両レバー:前方→前進

２. 両レバー:後方→後進

３. 両レバー:右方向→右方向移動

４. 両レバー:左方向→左方向移動

５. 両レバー:右斜め前方向→右斜め前方方向

６. 両レバー:左斜め前方向→左斜め前方方向

７. 両レバー:右斜め後方向→右斜め後方方向

８. 両レバー:左斜め後方向→左斜め後方方向

９. 右レバー:前方方向･左レバー:後方方向→左回転
１０. 右レバー:後方方向･左レバー:前方方向→右回転
１１. 右レバー:右方向･左レバー:左方向→×
１２. 右レバー:左方向･左レバー:右方向→×

３. 終了方法
3-１. 緊急停止ボタンを押す
3-２. Key をまわす

以上。

２-４. 全方向移動型歩行訓練機仕様書

部 品	型番	数量
UPP ボード	JPC20-UPP	1
A/D・D/A ボード	T104-ADA(TAC INC)	1
CPU ボード	JPC10-25V	1
電源ボード	JPC-POWER	1
PWM ボード	JPC20-UPP-PWMDRV	1
モータドライバ	TITECH Robot Driver PC0121-1	4
BATTERY	DC12V-7Ah	5

表(2.1)

以下に今回使用した、ボードの紹介をしておく。

1.UPP ボード



図(2.8)

この UPP ボードは、日立のユニバーサルパルスプロセッサを搭載しており、10 ビット A/D 変換 10 CH に加え、パルス入力、パルス出力、また PWM 制御などの 15 種類のコマンドを組み合わせることにより複雑なパルス制御を自動的に行わせることができるので、CPU の負担を大幅に軽減できる。

次ページに、UPP ボードの仕様を示す。

UPP ボード仕様

UPP	**日立 HD63140CP クロック 4MHz**
アナログ入力	**AN0～AN9の合計10CH** **分解能 10ビット、変換速度42μ秒／CH**
パルス入出力	**UU0～UU15の合計16CH** **入出力方向は8CH単位で切り替え可能** **標準 UU0～UU7 ／出力** **UU8～UU15／入力**
	出力方式 オープンコレクタ出力 **耐圧30V 電流100mA** **連続300mA（短時間）** **LED 「1」出力でLED点灯** **応答周波数 MAX 500KHz**

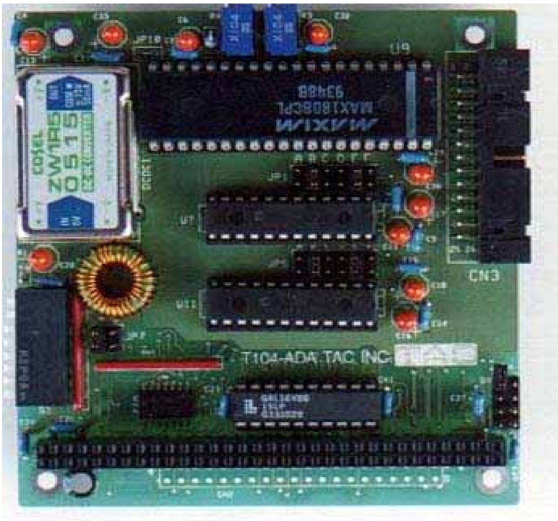
	入力方式　　電流駆動入力 入力抵抗　　４７Ω ＬＥＤ　　　負荷側ＯＮでＬＥＤ点灯 応答周波数　ＭＡＸ　５００ＫＨｚ
割り込み（ＵＰＰ内蔵）	ＩＮＴ０　パルス信号のエッジ割り込み ＩＮＴ１　Ａ／Ｄ変換の終了割り込み （それぞれ、任意のＩＲＱ割り込みに割り当て可能）
ウォッチドグタイマ	リセットサイクル　０．１２８ms～１３１ms（ＵＰＰ内蔵）
ボードサイズ	９６mm×９０mm×２５mm
重量	約７５ｇ
消費電流	５Ｖ±５％　１００mA（標準） １２Ｖ±５％　１００mA（標準）
使用温度範囲	０～６０℃

特徴

- JPC20-UPP は、PC/104 規格に準拠している。
- システムに応じ、必要なボードを CPU ボードに重ねていけるので組み込みスペースが小規模で済む。
- パルス信号の立ち上がり、立ち下がりエッジでの IRQ 割り込みや、A/D 変換の終了で IRQ 割り込みを発生させることが可能

※ PC/104 規格とは、DOS/V マシンの ISA バスを 64 ピンと 40 ピンのコネクタに変換したもののこと

2.A/D・D/A ボード



図(2.9)

T104-ADA 図(2.9)は株式会社ティーエーシーの製品であり、PC/104 バス対応の A/D+D/A 変換モジュールで、基盤に 12 ビット A/D コンバーターと 12 ビット D/A コンバーターを搭載している。

仕様

A/D コンバーター

分解能	12bit
入力チャンネル数	8ch 差動入力時:4ch
変換時間	8μsec 変換終了で割り込み可
入力電圧範囲	0〜5V、 0〜10V、 -2.5V〜+2.5V、 ch ごとにソフトで設定可

D/A コンバーター

分解能	12bit
出力チャンネル数	4ch
セットリング時間	8μsec max
出力電圧範囲	05V、 010V、 -5V+5V、 ch ごとにジャンパーピンで設定可

その他

基盤サイズ	90.1mm×95.8mm

電源	+5V、 320mA

特徴

- 90.1mm×95.8mm のモジュールを重ね合わせるだけ
- 省電力 1～2W/1 モジュール
- 低コスト
- PC/104 に準拠

3. CPU ボード



図(2.10)

CPU ボードは JPC10-V25CPU ボード図(2.10)を使用した。この製品は、日本システムデザイン(株)で作成されている。

仕様

<table>
<tr><td>メーカ</td><td colspan="2">日本システムデザイン(株)</td></tr>
<tr><td>型番</td><td colspan="2">JPC10-v25</td></tr>
<tr><td rowspan="4">スペック</td><td>CPU</td><td>μPD70352GJ-10
外部クロック 19.6608 MHz
内部クロック 9.8304 MHz</td></tr>
<tr><td>EEPROM</td><td>HN58C1001P-15</td></tr>
<tr><td>RAM</td><td>512K バイト擬似 SRAM</td></tr>
<tr><td>Size</td><td>96mm × 90mm × 15mm</td></tr>
<tr><td>入出力ポート</td><td colspan="2">8Ch (フラットケーブルコネクタ)</td></tr>
<tr><td>入力ポート</td><td colspan="2">4Ch (PC/104 バス)</td></tr>
<tr><td>コンパレータ入力</td><td colspan="2">8Ch (フラットケーブルコネクタ)</td></tr>
<tr><td>シリアルポート</td><td colspan="2">Ch0: RS-485 レベル</td></tr>
</table>

	Ch1: RS-232C レベル 通信速度: 110～115、 2Kbps
ポートサイズ	96mm×90mm×15mm
重量	約 75g
消費電流	5V ±5% 200Ma (標準)

特徴

・ PC104 バス規格に準拠

4. 電源ボード



図(2.11)

電源ボードには、日本システムデザイン株式会社製の JPC-POWER を使用した。入力電圧は+14.4V の単電源で、+5V と +12V の電源を供給することが可能。

入力電源	田宮模型社製 1700SCR 相当品 7.2V – 1700mA ニッカドバッテリ×2 個 または、スイッチング電源 15V
出力電源	コーセル社製 ZUS151205 +5V 2.4A
フリーエリア	2.54mm ピッチ 約 25 個×10 個
電源スイッチ	フジソク社製 8G2014(2 極トグルスイッチ)
ヒューズ	125V 10A 相当品
ボードサイズ	109mm×90mm×20mm
重量	約 100g

使用温度範囲	0～60℃

特徴

- 電源用トグルスイッチを持っているので、電源のオン、オフが簡単
- 回路の保護用にヒューズの組み込み
- JPC シリーズのボード評価用に便利

5. PWM ボード

PWM ドライブボード(JPC20-UPP-PWMDRV)の調整

・ボリュームの調整

V201 CN205 に接続したロータリエンコーダのレベル調整
ロータリエンコーダを回しながら、 JPC20-UPP ボード上の LED[UU8]と[UU9]が均等に点滅するよう調整

V202 CN206 に接続したロータリエンコーダのレベル調整
ロータリエンコーダを回しながら、JPC20-UPP ボード上の LED[UU10]と[UU11]が均等に点滅するように調整

PWM ドライブボード(JPC20-UPP-PWMDRV)と UPP の関係

PWM ボード(JPC20-UPP-PWMDRV) と UPP ボード(JPC20-UPP) を標準的な接続をした場合、UPP のパルス入出力と各モータやエンコーダは表(2.1)の接続になる。

パルス入出力端子	PW ドライブボードでの機能	標準接続先
UU0	CN101 コネクタに接続されたモータの PWM/ENABLE 出力	ステアリングモータ
UU1	CN101 コネクタに接続されたモータの DIR/PWM 出力	ステアリングモータ
UU2	CN102 コネクタに接続されたモータの PWM/ENABLE 出力	未使用
UU3	CN102 コネクタに接続されたモータの DIR/PWM 出力	未使用
UU4	CN103 コネクタに接続されたモータの PWM/ENABLE 出力	右走行モータ
UU5	CN103 コネクタに接続されたモータの DIR/PWM 出力	右走行モータ
UU6	CN104 コネクタに接続されたモータの PWM/ENABLE 出力	左走行モータ
UU7	CN104 コネクタに接続されたモータの DIR/PWM 出力	左走行モータ
UU8	CN205 コネクタに接続されたモータの A 相入力	右ロータリエンコーダ
UU9	CN205 コネクタに接続されたモータの B 相入力	右ロータリエンコーダ
UU10	CN206 コネクタに接続されたモータの A 相入力	左ロータリエンコーダ

UU11	CN206 コネクタに接続されたモータの B 相入力	未使用
UU12	CN207 コネクタに接続されたモータの A 相入力	未使用
UU13	CN207 コネクタに接続されたモータの B 相入力	未使用
UU14	CN208 コネクタに接続されたモータの A 相入力	未使用
UU15	CN208 コネクタに接続されたモータの B 相入力	未使用

表(2.1)

PWM 出力は、モータの回転方向を指定して単方向の PWM を出力する方法と、正転/逆転を常に切り替える方法(つまりデューティー 50%で停止)が選択できる。

CN101 に接続されたモータを例にすると、
1.UU0 に PWM パルスを出力するとデューティ 100%が停止になり UU1 が回転方向の切り替えとなる
2.UU1 に PWM パルスを出力するとデューティ 50%が停止になり UU0 はモータイネーブル(LOW で ON、 HI で OFF)となる

UPP にはロータリエンコーダの 2 相パルスを 4 逓倍してアップダウンカウントする機能があります。この機能を使用したとき、走行距離 0.2mm/pulse の分解能となる

PWM ドライブボード(JPC20-UPP-PWMDRV)、UPP ボード(JPC20-UPP)の標準的な接続

UPP のパルス入出力と各モータやエンコーダは表(2.2)のような接続になる

アナログ入力端子	PWM ドライブボードでの機能	標準接続先
AN0	CN101 コネクタに接続されたモータの平均電流値入力	ステアリングータ電流値
AN1	CN102 コネクタに接続されたモータの平均電流値入力	
AN2	CN103 コネクタに接続されたモータの平均電流値入力	右モータ電流値
AN3	CN104 コネクタに接続されたモータの平均電流値入力	左モータ電流値
AN4	CN201 コネクタに接続されたポテンションメータの値入力	ステアリングポテンションメータ
AN5	CN202 コネクタに接続されたポテンションメータの値入力	
AN6	CN203 コネクタに接続されたポテンションメータの値入力	
AN7	CN204 コネクタに接続されたポテンションメータの値入力	
AN8	モータ電源電圧入力	バッテリ電圧
AN9	未使用	

表(2.2)

モータ電流値は電圧に変換されて入力される。電流と電圧の関係は、0.57V/A となり、A/D コンバータは 10bit、入力範囲は 0 - 5V ですから変換後のデータが 117 の場合は 1A となる。

また、ステアリング位置を検出するポテンションメータは有効回転各 324 度で、その角度を 10bit 精度の A/D コンバータで読み込む。ステアリング位置と A/D 変換地の関係は表(2.3)に示しておく。

ステアリング位置	ポテンションメータ電圧	A/D 値
右 90 度	1.11V	227
直進	2.50V	512
左 90 度	3.89V	796

表(2.3)

バッテリ電圧は 1/5.7 に変換されて入力される。14.4V のバッテリ電圧時には 2.53V となり、A/D コンバータは 10bit、 入力範囲は 0-5V ですから変換後のデータが 518 の時 14.4V になる。

6. モータドライバ



図(2.12)

モータドライバには、TITECH ROBOT DRIVER 図(2.12)を使用した。

このドライバは(株)東京精機製で、知能機械システム全般で能動自由度の多様化が進んでいる現在、アクチュエータとコントローラ組のコストが全体の試作費に対して大きな割合

を占めている。そこでこれらの要素の低価格化が実現できれば、全体のコストを大幅に削減でき、多くの試作や、実験が可能となり、その結果機械システム全体の洗練と最適化が促進されるのではないかと期待される。このような事情で、全方向移動型歩行訓練機ではTITEC DRIVER と呼ぶ駆動回路を使用した。

仕様

項目	仕様	備考
定格出力電圧	±30V	
定格出力電流	±3.8A	
最大出力電圧	±45V	
最大出力電流 (連続)	±16.7A	
最大出力 (連続)	750W	十分な放射熱板使用時
主電源	DC12V～48V	
指令入力電圧	±10V	
外形寸法 (L W H)	100×90×35	
重量	130g	放射版無し

制御モード	電流制御 速度制御 位置制御	 タコジェネ不要 同上
保護機能	電流制限 (0～16.7A)	
モータ・ブレーキ機能	電気ブレーキ停止 フリー・停止 正逆転限界停止	モータ端子短絡 モータ端子解放 リミット・スイッチ使用

特徴

- 小型、軽量、高出力(750W)、 低自己消費型(PWM 駆動)
- 電流(トルク)制御、速度制御、位置制御、3 つのモードに対応
- タコジェネ無しで速度制御を実現 (「電子ガバナ回路」)
- タコジェネ無しで即応性のよい位置制御を実現
- ロボットの胴体や収納ケースを放熱器として使用可能

今回の全方向移動型歩行訓練機では速度制御を使用した。
次ページに、動作モード表(2.4)を示す。

動作モードの設定 (電子ガバナ回路使用)

調整部品	機能	速度制御
VT1	モータ電機子巻線抵抗値補償	◎
VT2	入力信号用零点調整	○
VT3	位置制御比例ゲイン(Kp)調整	□
VT4	速度フルケース(Kt)調整	○
VT5	電流制限/フルスケール調整	○
SW1	電流制御/位置速度モード切替	2-3
NFB 入力信号		OPEN

表(2.4)

◎ ⇒ 絶対に調整必要

○ ⇒ 調整可、動作に影響

□ ⇒ 半固定抵抗を左いっぱいに回す

OPEN ⇒ 開放しておくかグランドに接地する(外部信号を接続してはならない)

【速度制御モード】

速度制御モードでフィードバックに用いるモータ回転速度情報は、

(1) 擬似タコジェネ回路によって演算

(2) 外部の機械的タコジェネレータにより測定

【モータ電機子抵抗値 Ra の補償 (VT1)】

電子ガバナ回路を正常に動作させるには、使用するモータの電機子抵抗値 Ra に合わせて VT1 を調整する必要がある。そのためには Ra をモータの説明書から、あるいは実測して調べ、式(2.1)により VT1 で調整すべき抵抗値 VT1 res を計算する。

$$VT1res = \frac{500}{3 \cdot Nturns} Ra \quad [\mathrm{k}\Omega] \qquad 式(2.1)$$

VT1 は 0V～T1max[Ω]の範囲で調節可能なので、その回転量 VT1pos は式(2.2)のように計算できる。

$$VT1pos = \frac{VT1res}{VT1\max}$$ 式(2.2)

ただし、標準仕様では電流センサー一次側巻数は Nturns = 3、 半固定抵抗 VT1 は VT1max =200KΩです。

なお、VT1pos の設定は VT1 上に刻まれた印を基に目測で行いますので設定値に多少ずれが生じます。ここで注意すべきことはこの設定値を計算値より大きくした場合、回路の動作が不安定になることだ。解決策としては、計算値より少なめに設定することが挙げられる。

【モータ回転速度の計算】

電子ガバナ回路使用状態ではコネクタ CN2 の 4 番ピン(TG)からモータの回転速度ωpseudo をモニターできます。Ωpseudo[rpm]は TG 端子の出力電圧 Vpseudo[V]に式(2.3)のように直線的に比例。

$$\omega pseudo = \frac{Vrated}{300}\omega freeVpseudo$$ 【rpm】 式(2.3)

ただし、Vrated[V]はモータの定格電圧、ωfree[rpm]は定格電圧での無負荷回転数であり、対象とするモータの取扱説明書を参考のこと。

【速度フルスケール調整(VT4)】

最大入力指令電圧に対するモータ速度ωfs をフルスケール速度と呼び、ωfs は VT4 の回転量 VT4pos (0～100%)により式(2.4)のように設定。

$$\omega fs = \frac{47}{100VT4pos + 47}\omega pseudoMAX$$ 【rpm】 式(2.4)

ただし、ωpseudoMAX は式(2.2)において Vpseudo=10V とした回転速度である。

【電流センサ改造】

電流センサのフルスケール電流 Ismax と一次側巻数 Nturns の関係は式(2.5)のようになる。

$$Is\max = \frac{50}{Nturns}$$ 【A】 式(2.5)

Nturns は 3 より少なくしてはいけませんが、増やすことは可能。低電流の DC モータを駆動

する場合には Nturns を実際に使用する電流レンジに合わせることで信号/ノイズ比を向上できる。

7．BATTERY



図(2.13)

バッテリは、GS 社の 12V-7A/h を 5 つ、使用した。制御回路に一つと、各オムニホイルに一つずつである。

以上のような、パーツを「全方向移動型歩行訓練機」に使用した。

第3章　全方向移動制御

3.1 プログラム

ここで今回,使用したプログラムを記載しておく。

ヘッダーファイル

```
#define DEBUG_ON

#ifndef WALKER__H
#define WALKER__H

#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <dos.h>
#include <stdlib.h>

#include "v25.h"

//------------ T104-ADA base I/O address-----------begin
#define  BIO          0x300

/*D/A I/O address*/
#define DA1L    BIO+0x00
#define LDAC    BIO+0x00
/*A/D I/O address*/
#define ADL     BIO+0x08
#define ADU     BIO+0x09
#define ADSTS BIO+0x0a

//------------ T104-ADA base I/O address-----------end
```

```
#define    cls()                   printf("¥x1b[2J")
#define    locate(x,y)     printf("¥x1b[%d;%dH", y, x)
#define    clsxy(x,y)     printf("¥x1b[%d;%dH¥x1b[K",y,x)

#define    Rad2Deg       57.2958
#define    Deg2Rad       0.01745
#define    PI           3.1415926

void      iniz_port2(void);

float ad_data1(int ch);
float     ad_data8(int ch);
void      isetda(int ch,int data);
void      Set0_Motor(void);
void   wait(int time);

//flag=0    wait for enter;   flag=1   detect if enter.
unsigned char Getch(int flag);

void JoyTest(void);
double JoyDetect1(void);
double JoyDetect2(void);
void DATest(void);
void MotionTest(void);
void OutSpeed(double vx, double vy, double ee, int *Stop);

#endif
```

以上。示したヘッダーファイルを使用して,次に示すプログラムを動かしている。

Main 関数プログラム

```
Walker.0
#include "walker.h"

void main(void)
{
        int ch;

        iniz_port2();
        Set0_Motor();

#ifdef DEBUG_ON
        do{
                cls();
                printf("¥n --------------------Main Menu-------------------¥n¥n");
                printf(   "                     1 -------- Joystick test¥n");
                printf(   "                     2 -------- Test D/A¥n");
                printf(   "                     3 -------- Motion Test¥n");
                printf(   "                   <Esc>------- Terminate program!¥n");
                printf(   " ----------------------------------------------¥n");
                printf(   " Please choose:");
                ch=getch();

                cls();

                switch(ch){
                        case   '1':
                                JoyTest();
                                break;
                        case   '2':
                                DATest();
                                break;
                        case   '3':
```

```
#endif
                                        MotionTest();
#ifdef DEBUG_ON
                                        break;
                    }
                    Set0_Motor();
          }while(ch!=0x1b);
#endif

}
```

歩行器プログラム

```
#include "walker.h"

void iniz_port2(void)                    // CN202 の初期化
{
          pokeb(_V25BASE,_PMC2,0x00);     // ポート 2 をﾎﾟｰﾄﾓｰﾄﾞに設定
          pokeb(_V25BASE,_PM2, 0x00);     // 出力ポートに設定
}

void isetda(int ch,int data)         //ch(1--4),data(0--0x0fff) モータへの命令
{
          if (ch<=0||ch>4)    return ;

          if (data<0)    data=0;
          else if (data>0xfff) data=0xfff;

          outp((DA1L+(ch*2)-2),data&0xff);
          outp((DA1L+(ch*2)-1),data>>8);
          inp(LDAC);
}

float ad_data1(int ch)      //    1--8
{
```

```
        float x;

        outp(ADL,ch-1);     // select ch.& convert start
        while((inp(ADSTS)&0x80)==0) { ; }
        x=((inp(ADU)<< 8)+inp(ADL))*5/4096.0;
        return(x);
}

float ad_data8(int ch)      //  1--8
{
        float sum=0;
        int i;

        ch--;
        for (i=0;i<6;i++) {
                outp(ADL,ch);    // select ch.& convert start
                while((inp(ADSTS)&0x80)==0) { ; }
                sum+=inp(ADL)+(inp(ADU)<< 8) ;
        }
        sum=sum/6.0*5.0/4096.0;
        return sum;
}

void  Set0_Motor(void)
{
        isetda(1,2048);
        isetda(2,2048);
        isetda(3,2048);
        isetda(4,2048);
}

//flag=0   wait for enter;  flag=1  detect if enter.

unsigned char Getch(int flag)
{       unsigned  char ch;
```

```
        switch(flag){
                case 0:
                        while(!kbhit()) ;
                        ch=getch();
                        if(ch==0)   ch=getch();
                        break;
                case 1:
                        if(kbhit()){
                                ch=getch();
                                if(ch==0)   ch=getch();
                        }
                        else   ch=0;
                        break;
        }
        return ch;
}

double JoyDetect1(void)
{
        float xDelta=0.5,yDelta=0.5,dxDelta=0.05,dyDelta=0.05;
        static float xOld=0,yOld=0,xNew=0,yNew=0,dx,dy;
        static double xyAngle=888;

        xOld=xNew;
        yOld=yNew;

        xNew=ad_data8(1)-2.25;
        yNew=ad_data8(2)-2.15;

        if((fabs(xNew)<xDelta) && (fabs(yNew)<yDelta)){
                xyAngle=888;

                return xyAngle;
        }

        dx=xNew-xOld;
```

```
        dy=yNew-yOld;

        if((fabs(dx)<dxDelta) && (fabs(dy)<dyDelta)){
                xNew=xOld;
                yNew=yOld;
                return xyAngle;
        }

        xyAngle=Rad2Deg*atan2(yNew,xNew);
        if(fabs(xyAngle+180)< 0.01) xyAngle=180;
        return xyAngle;
}

double JoyDetect2(void)
{
        float xDelta=0.5,yDelta=0.5,dxDelta=0.05,dyDelta=0.05;
        static float xOld=0,yOld=0,xNew=0,yNew=0,dx,dy;
        static double xyAngle=888;

        xOld=xNew;
        yOld=yNew;

        xNew=ad_data8(3)-2.49;
        yNew=ad_data8(4)-2.50;

        if((fabs(xNew)<xDelta) && (fabs(yNew)<yDelta)){
                xyAngle=888;
                return xyAngle;
        }



        dx=xNew-xOld;
        dy=yNew-yOld;

        if((fabs(dx)<dxDelta) && (fabs(dy)<dyDelta)){
                xNew=xOld;
```

```
		yNew=yOld;
		return xyAngle;
	}

	xyAngle=Rad2Deg*atan2(yNew,xNew);
	if(fabs(xyAngle+180)< 0.01) xyAngle=180;
	return xyAngle;
}

void JoyTest(void)
{
	float xNew[2],yNew[2],k;

	printf("¥n Joystick test:¥n");
	printf(" Press any key and break...¥n");

	while(!kbhit()){

		xNew[0]=ad_data8(1);
		yNew[0]=ad_data8(2);
		xNew[1]=ad_data8(3);
		yNew[1]=ad_data8(4);
		k=ad_data8(5);

		locate(1,8);
		printf(" Left: xNew=%5.2f	yNew=%5.2f	angle=%6.1lf¥n¥n",
			xNew[0],yNew[0],JoyDetect1());
		printf(" Right: xNew=%5.2f	yNew=%5.2f	angle=%6.1lf¥n¥n",
			xNew[1],yNew[1],JoyDetect2());
		printf("Potentiometer:	%5.2f¥n",k);
	}
	getch();
}

void DATest(void)
{ int ch=0,chan,data;
```

```
		locate(1,4);
		printf("DA test:¥n");

		while(ch!=0x1b){
			clsxy(1,8);
			printf("Please input channel(1-4),data(-2048~~2048):");
			scanf("%d,%d",&chan,&data);
			isetda( chan,(int)(data+2048) );
			printf("¥n¥nPress <ESC> break and other key continue...");
			ch=getch();
			Set0_Motor();
		}
}

void MotionTest(void)
{
   double xyAngle[2],Speed,vx=100,vy=100,ee=0,err=0,errRange=10;
   double dtemp;
   int Stop=0;

#ifdef DEBUG_ON
		locate(1,4);
		printf("Motion test:¥n");
		printf("Press <ESC> exit...¥n");

#endif

		while(Getch(1)!=0x1b){
			xyAngle[0]=JoyDetect1();
			xyAngle[1]=JoyDetect2();

			if( xyAngle[0]>800 || xyAngle[1]>800){
				if(Stop==1 || Stop==2) Stop=3;
				else if(Stop==0)			vx=vy=ee=0;
			}else{
```

```
                        if(Stop==0) Stop=1;
                        Speed=100*ad_data8(5)/5.0;

                        err=xyAngle[0]-xyAngle[1];
                        if(abs(err)<errRange){
                                dtemp=(xyAngle[0]+xyAngle[1])/2;
                                vx=Speed*cos(dtemp/180*3.1415);
                                vy=Speed*sin(dtemp/180*3.1415);
                        }else if( abs(err+360)<errRange ){
                                dtemp=(xyAngle[0]+xyAngle[1]+360)/2;
                                vx=Speed*cos(dtemp/180*3.1415);
                                vy=Speed*sin(dtemp/180*3.1415);
                        }else if( abs(err-360)<errRange ){
                                dtemp=(xyAngle[0]+xyAngle[1]-360)/2;
                                vx=Speed*cos(dtemp/180*3.1415);
                                vy=Speed*sin(dtemp/180*3.1415);
                        }else if( abs(abs(err)-180)<errRange){
                                if(err>0) ee=-100;
                                else      ee= 100;
                        }

                }

                OutSpeed( vx,vy,ee,&Stop);

        }

}

void OutSpeed(double vx, double vy, double ee, int *Stop)
{ int i;
        double M=1.0,Period;
        static int nStart,nStop;
        static int v[4]={0,0,0,0};

        if(*Stop==0){
```

```
			nStart=nStop=0;
		}

		v[0]= vx-vy+ee;
		v[1]=-vx-vy+ee;
		v[2]=-vx+vy+ee;
		v[3]= vx+vy+ee;

		M = 1.0;
		if( *Stop==1){
			Period=100;
			if ( nStart<=Period ) {
				M = 0.5 * ( 1 - cos (PI * nStart / Period ) );
				nStart ++;
			}else{
				*Stop=2;
			}
		}

		if ( *Stop==3 ) {
			Period=10;
			if ( nStop > Period) {
				*Stop=0;

		return ;
			}
			M *= 0.5 * ( 1 + cos (PI*nStop / Period) );
			nStop ++;
		}

		for(i=0;i<4;i++)
			isetda( i+1,(int)(v[i]*M+2048) );

#ifdef DEBUG_ON
		locate(1,10);
		printf("Stop=%3d
```

```
M=%2.1lf¥n%4d %4d %4d %4d¥n",Stop,M,v[0],v[1],v[2],v[3]);
#endif
}
```

3.2 プログラムの説明

ヘッダーファイル

#define DEBUG_ON→この関数により全てのヘッダーファイルを有効にする。

#include “v25.h”→v25.h をインクルードする

```
#define BIO       0x300

/*D/A I/O address*/
#define DA1L  BIO+0x00
#define LDAC  BIO+0x00
/*A/D I/O address*/

#define ADL   BIO+0x08
#define ADU   BIO+0x09

#define ADSTS BIO+0x0a
```

↓

T104-ADA base I/O adress の定義

#define cls() printf(″¥x1b[2J″)→cls 関数の定義

#define locate(x,y) printf(″¥x1b[%d;%dH″, y, x)→locate(x,y) 関数の定義

#define clsxy(x,y) printf(″¥x1b[%d;%dH¥x1b[K″,y,x)→clsxy(x,y) 関数の定義

#define Rad2Deg 57.2958→Rad2Deg を 57.2985 に定義

#define Deg2Rad 0.01745→Deg2Rad を 0.01745 に定義

#define PI 3.1415926→PI を 3.1415926 に定義

void iniz_port2(void);→関数の呼び出し

float ad_data1(int ch);→関数の定義
float ad_data8(int ch);→関数の定義

void isetda(int ch , int data);→関数呼び出し

void Set_Motor(void);→関数呼び出し

void wait(int time);→関数呼び出し

unsigned char Getch(int flag);→flag(0〜255)の入力

void JoyTest(void);→関数呼び出し
double JoyDetect1(void);
double JoyDetect2(void);

void DATest(void);→関数呼び出し

void Out Speed(double vx, double vy, double ee, int *Stop);→関数呼び出し

全方向移動型歩行訓練機プログラム
xyAngle[n]→2つの Joysutick の角度

err=xyAngle[0]-xyAngle[1];→xyAngle[0],[1]の誤差を出力

dtemp=(xyAngle[0]-xyAngle[1])/2→中位置検出(-10°〜 10°) ※1
※1. L(左)と R(右)の誤差が 10°以内なら、間の 5°の方向に進む。
10°以上の誤差があれば停止

次ページに図で示す。

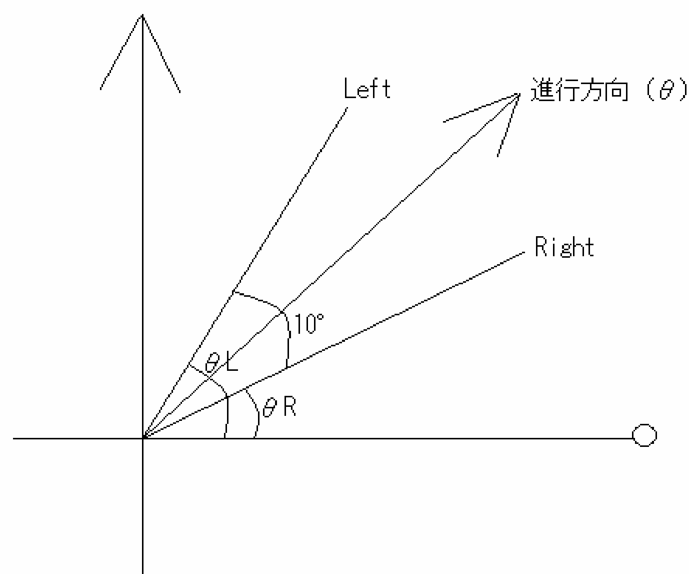


図 3.1

直線移動

・dtemp=(xyAngle[0]-xyAngle[1])/2→目標角度検出 -10°≦ err ≦ 10°

・dtemp=(xyAngle[0]-xyAngle[1]+360°)/2→目標角度検出 -360°≦ err ≦ -350°

・dtemp=(xyAngle[0]-xyAngle[1]-360°)/2→目標角度検出 350°≦ err ≦ 360°

図 3.1 のθL は左ジョイスティック・θR は右ジョイスティックまでの角度を表し，Left，Right は左右･ジョイスティックの倒す方向を示し，進行方向(V)は $V=\frac{\theta L+\theta R}{2}$ 方向に進む。

回転

abs(abs(err)-180°) ＜ errRange→回転 ※2

※2.

err→左右 Joystick の角度差

errRange→左右 Joystick の誤差の範囲

左右のジョイスティックをそれぞれ反対方向に倒すことで回転が可能で，回転することにより歩行訓練機の周り 360°全方向に姿勢を変化させることが可能である。

ad_data8(5)→信号(速度調整のポテンショメータ) 0～5V

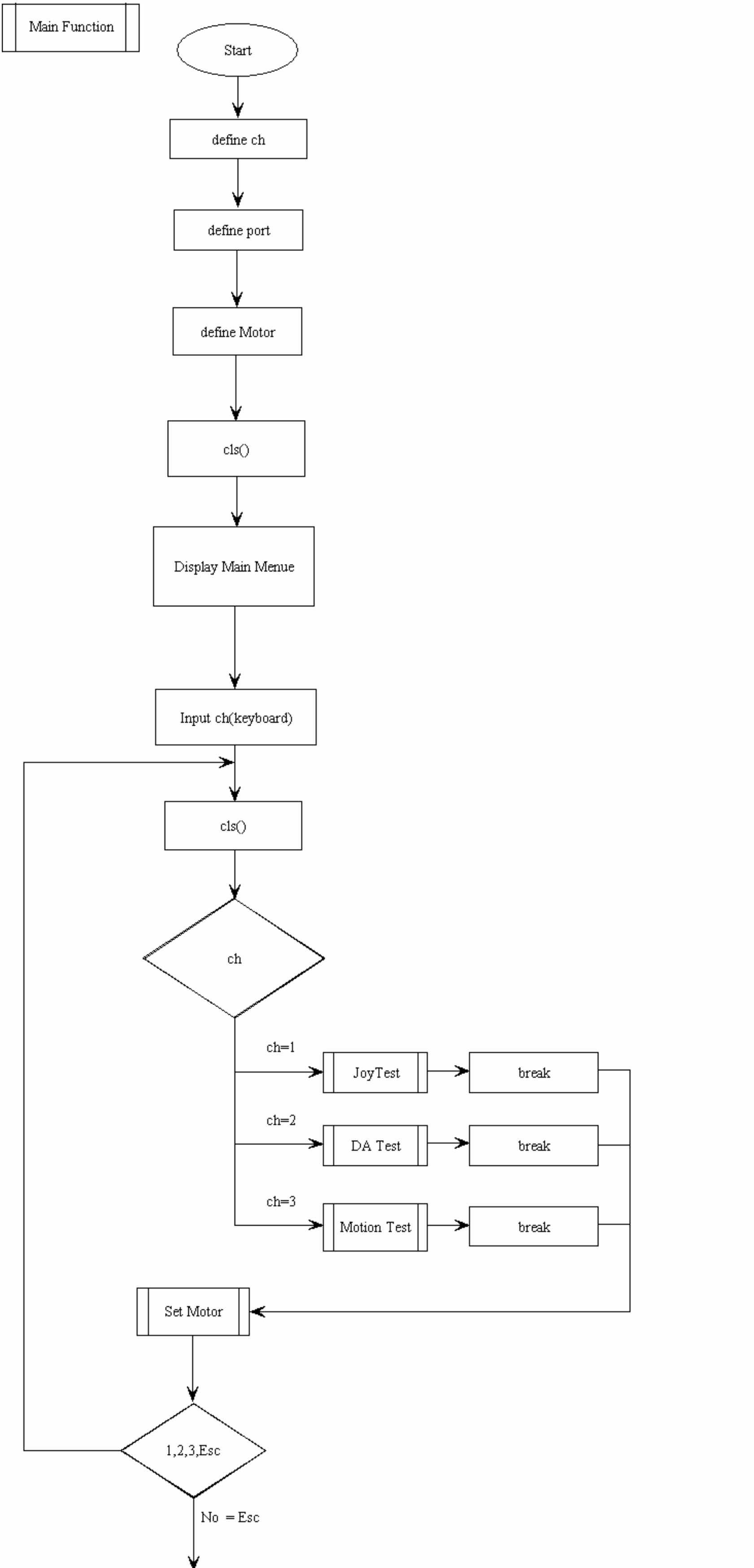
3.3 フローチャート

プログラムのフローチャートを示す。

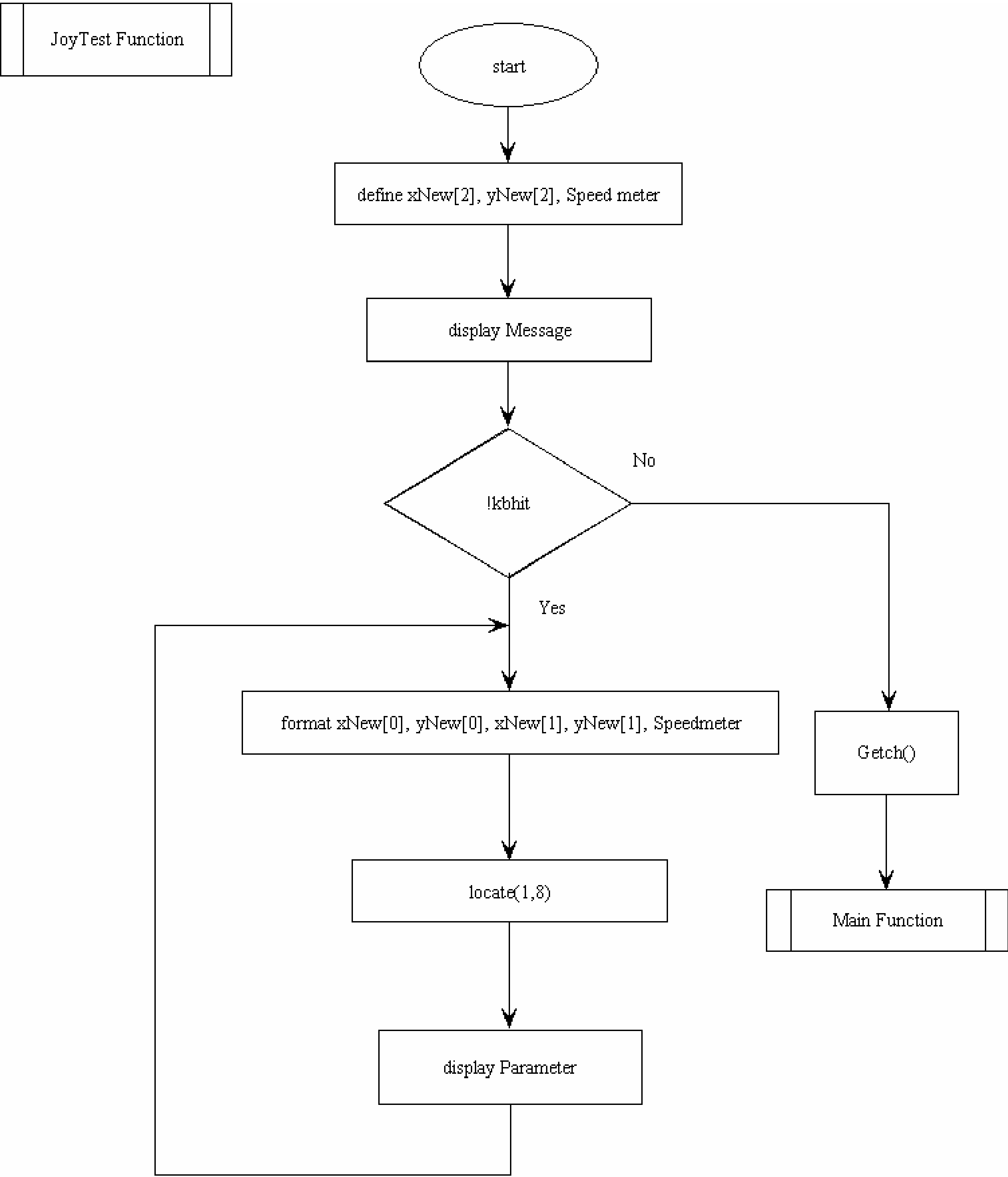
フローチャート一覧表

1.Main 関数
2.ジョイスティック関数
3.Test D/A 関数
4.Motion Test 関数
5.OutSpeed 関数

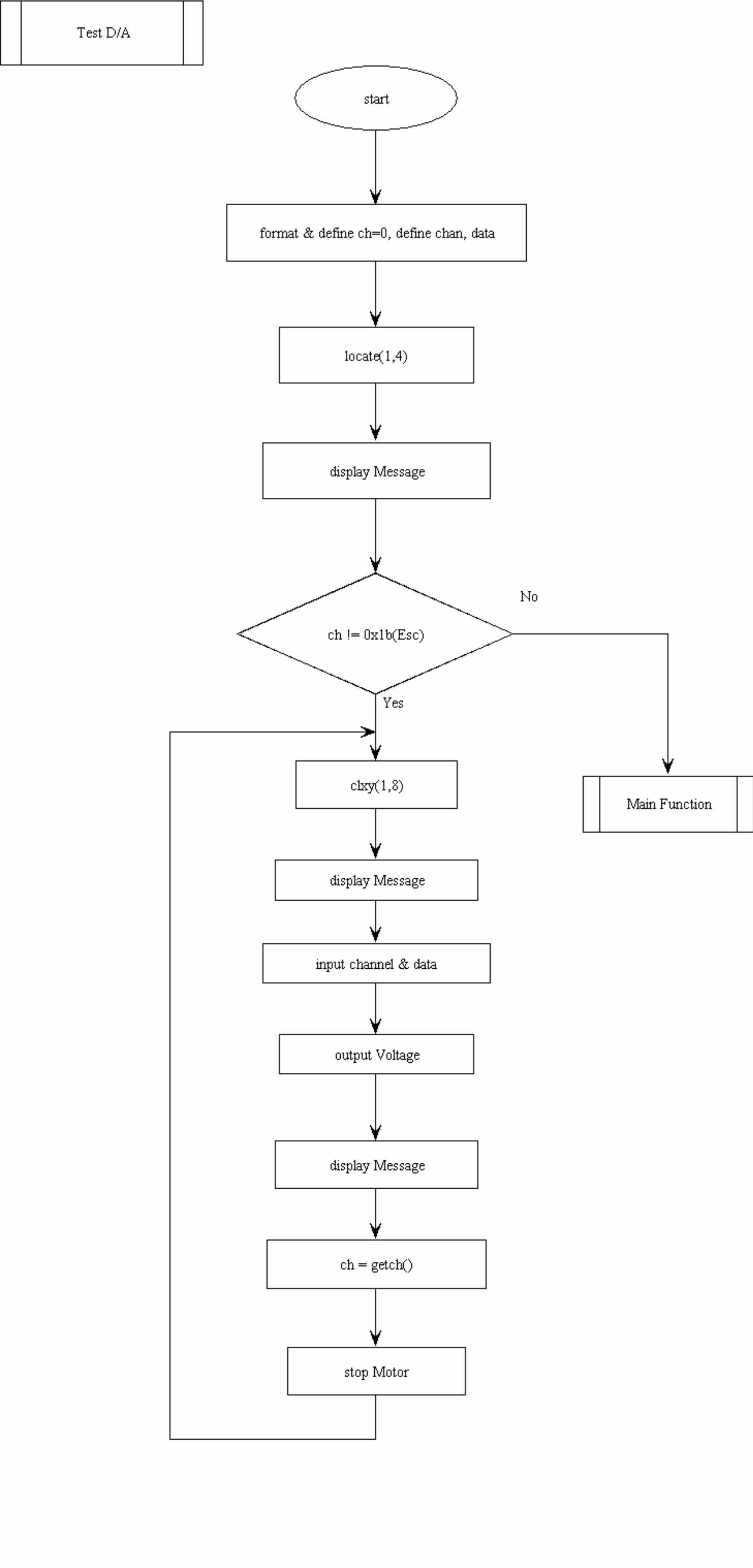
以上のようにフローチャートを示している。

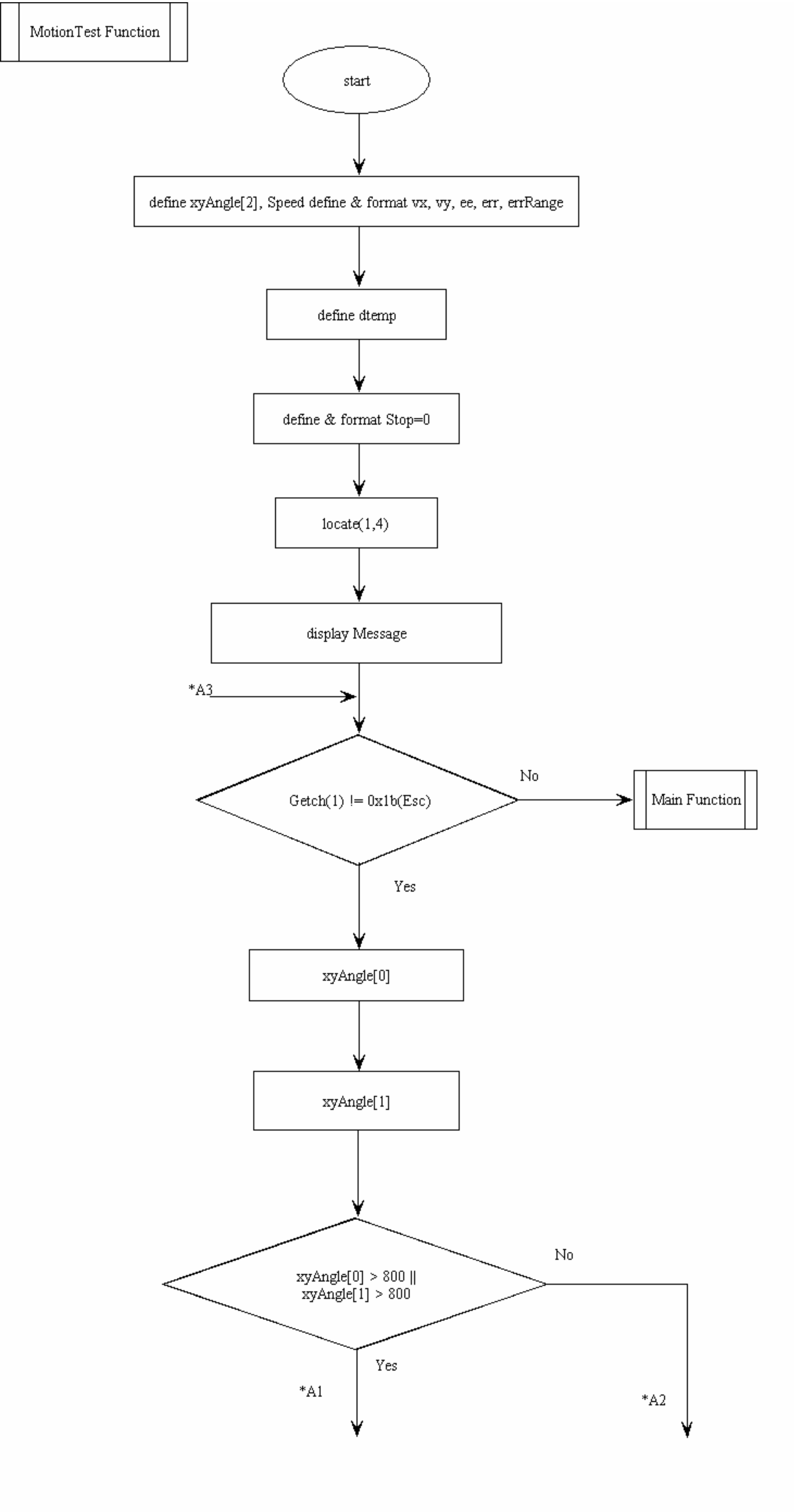

Main Function
Start
define ch
define port
define Motor
cls()
Display Main Menue
Input ch(keyboard)
cls()
ch
ch=1
JoyTest
break
ch=2
DA Test
break
ch=3
Motion Test
break
Set Motor
1,2,3,Esc
No =Esc

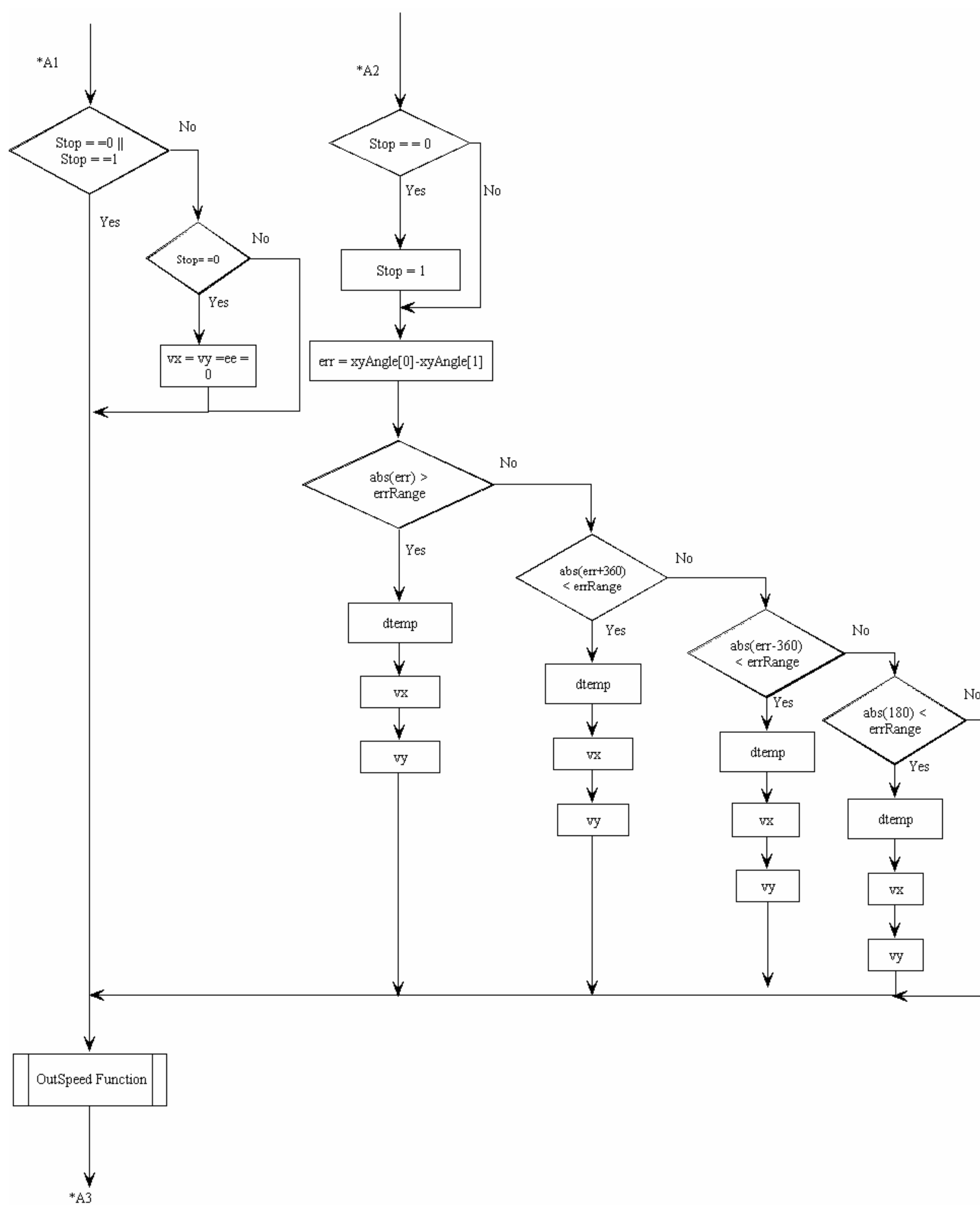
Joytest 関数 Flowchart



次ページに、D/Atest 関数の Flowchart を載せておく。

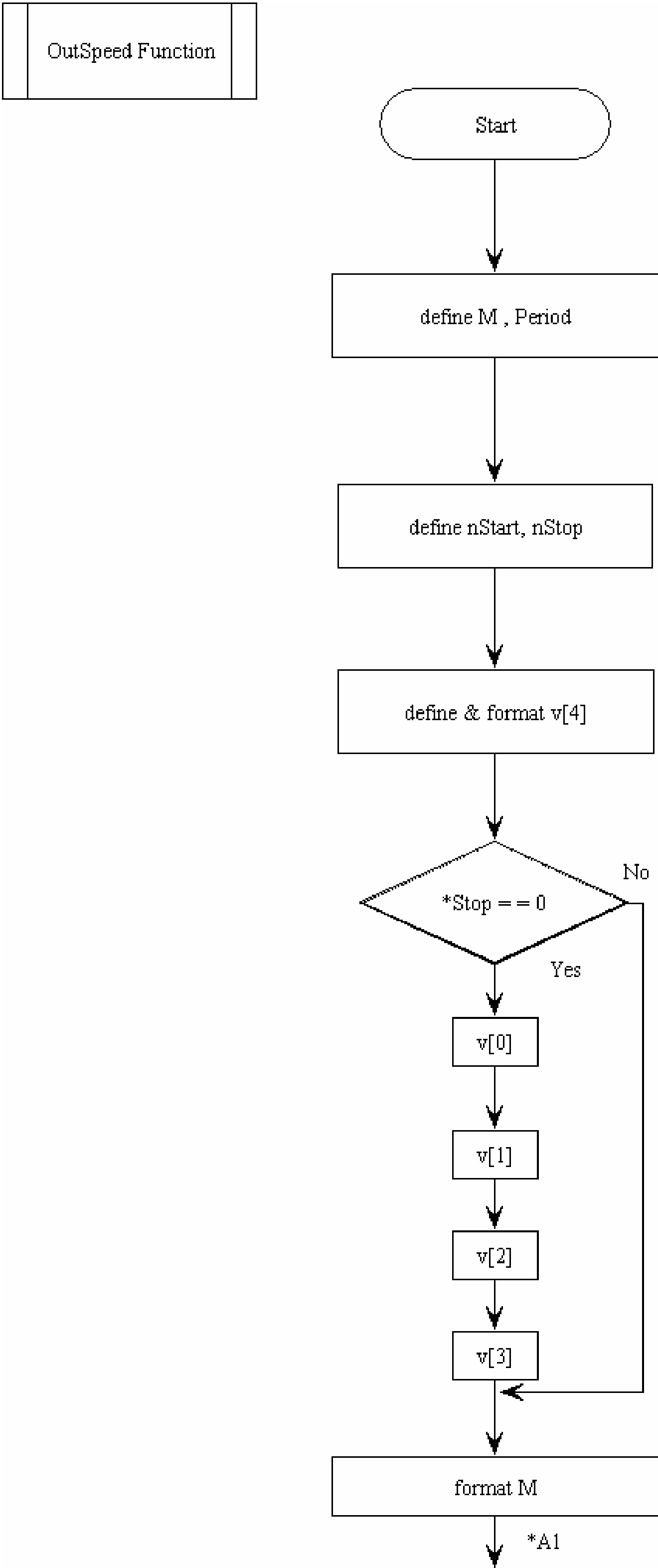

Test D/A
start
format & define ch=0, define chan, data
locate(1,4)
display Message
ch != 0x1b(Esc)
No
Yes
clxy(1,8)
Main Function
display Message
input channel & data
output Voltage
display Message
ch = getch()
stop Motor

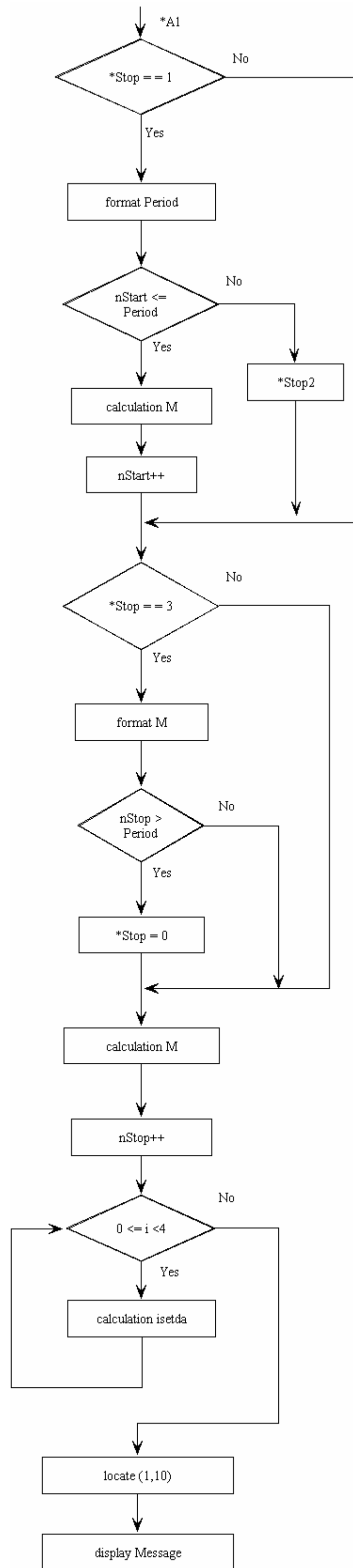

MotionTest Function
start
define xyAngle[2], Speed define & format vx, vy, ee, err, errRange
define dtemp
define & format Stop=0
locate(1,4)
display Message
*A3
Getch(1) != 0x1b(Esc)
No
Main Function
Yes
xyAngle[0]
xyAngle[1]
xyAngle[0] > 800 ||
xyAngle[1] > 800
No
Yes
*A1
*A2



以上のフローチャートが、 MotionTest 関数である。

次ページに、OutSpeed 関数を示しておく。


OutSpeed Function
Start
define M , Period
define nStart, nStop
define & format v[4]
*Stop = = 0
No
Yes
v[0]
v[1]
v[2]
v[3]
format M
*A1


*A1
*Stop = = 1
No
Yes
format Period
nStart <=
Period
No
Yes
calculation M
*Stop2
nStart++
*Stop = = 3
No
Yes
format M
nStop >
Period
No
Yes
*Stop = 0
calculation M
nStop++
0 <= i <4
No
Yes
calculation isetda
locate (1,10)
display Message

第 4 章 実験と考察

この章では私が行った、実験のことを記しておく。

私は、以前の「全方向移動型歩行訓練機」のスピードでは、お年寄りが使うには幾分歩行訓練機のスピードが速すぎて、お年寄りは着いていけないのではないかと考え、スピードの調整と、発進時の急なスピードの上昇、そして停止時にジョイスティックを戻してもその場では停止せずに、少し進んで停止するという誤差が出てしまう。その点の改善を目指した。

先ず歩行訓練機のスピードの速さであるが、これはスピード調整のプログラムを変更することにより解決することができた。その詳細は、

$$Speed = 300 * ad_data8(5) / 5.0 \quad \text{式(4.1)}$$

式(4.1)を式(4.2)のように変更することで解決することができた。

$$Speed = 100 * ad_data8(5) / 5.0 \quad \text{式(4.2)}$$

こうすることにより直接歩行訓練機のスピードを落とし、スピードを調節することが可能になった。

また、次は歩行訓練機の急発進、誤差のでる停止であるが、その前に少し歩行訓練機のプログラムの解説を加えておく。

第 3 章で紹介したプログラムソースの内で、

Stop == n

Stop == 0;　　停止状態

Stop == 1;　　加速状態

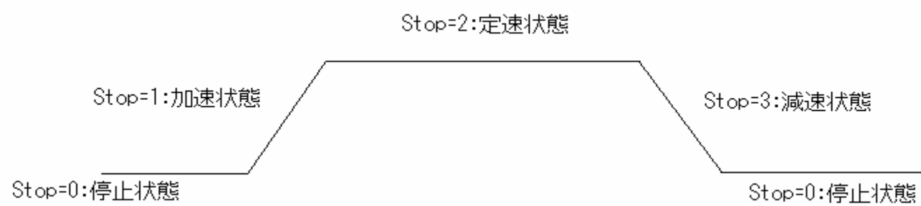
Stop == 2;　　定速状態

Stop == 3;　　減速状態

以上のような、文がある。

Stop == n (0～4)の状態により、そのときの歩行訓練機の動作を表している。

(次ページの図(4.1)参照)



図(4.1)

そこで、Stop==n (0～4)までの値を直接代入する方法を考え実行した。
以下のソースがその際に使用したソースである。

```
i=0;
if(Stop == i)
        switch(i){
                case 1:
                        Speed=100*ad_data8(5)/5.0;
                case 2:
                        Speed=150*ad_data8(5)/5.0;
                case 3:
                        Speed=50*ad_data8(5)/5.0;
        }
```

この状態でも歩行訓練機は走るには走ったがしたが、スピードのコントロールができず常に、一定の速度のままで走ってしまい求めている結果とは違う結果がでてしまった。

結論として、上記のプログラムは使用不可能。
そこで、次ページに示すプログラムを使用した。

このプログラムは、式(4.3)の Period の値を直接代入することにより、立ち上がり、立下りの曲線を二次曲線にする事により、スピードを調節しようと考えた。

$$M = 0.5*(1-\cos(PI*nStart/Period))$$
$$M* = 0.5*(1+\cos(PI*nStop/Period))$$ 式(4.3)

※ M＊＝は M＊1 のこと

下記にプログラムのソースを示す。

```
M=1.0;
If(*Stop==1){
        Period=100;
        if(nStart<=Period){
                M=0.5*(1-cos(PI*nStop/Period));
                nStart++;
        }else{
                *Stop=2;
                }
        }
        if(nStop>Period){
                        *Stop=0;
                        return ;
                }
                M*=0.5*(1*cos(PI*nStop/Period));
                nStop++;
        }
```

こちらのプログラムは発進時には緩やかに発進でき、また停止時にもその場で停止することができた。
以上の結果より、実験は成功した。

第 5 章 まとめ

この章では、私の実験した全方向移動型歩行訓練機に対する私の考えを示しておく。

5.1 本研究の成果

全方向移動型歩行訓練機のプログラムは今回の研究により実現できた。

5.2 今後の研究課題

今回の研究は全方向移動方歩行訓練機の一端に触れたに過ぎないと私は考えている。

そこでより深いところまで、全方向移動型歩行訓練機を理解し、より一層高齢者の使えるものが必要になると考える。
今後の課題は、先ず全方向移動型歩行訓練機を完全に理解し、より良い全方向移動型歩行訓練器のプログラムの作成と、オムニホイルの特性を調べ、自分の行きたい方向への移動ができているかを調べること、また実際の施設の訪問なども考えている.

以上。

謝辞

　本研究を行うにあたったて終始ご指導下さった高知工科大学知能機械システム工学科王碩玉教授に対し深く感謝いたします。

　また，本研究を行うに対して助言をして下さった陳貴林氏に対して深く感謝いたしております。

参考文献

[1]RPS 事業だより:http://www.joho-kochi.or.jp/johosi/0311/rsp.html

[2]天井吊り下げ式歩行支援システム「フローラ」:
http://www1.odn.ne.jp/~aae76220/flora.pdf

[3]川崎医療福祉学会誌:Vol.13 No.1 2003 63-67，重度障害児用歩行器 SRC ウォーカーの適合と評価ー療育施設と養護学校の事例を通してー:
http://shuttle.kawasaki-m.ac.jp/mw/Journal_HP/jjournal/2003_J13-1/07-shigenari.pdf

「4」卒業論文集 高知工科大学，知能機械システム工学科，ジョイスティックによる全方向移動車の制御，2001

「5」卒業論文集 高知工科大学，知能機械システム工学科，転倒防止できる全方向移動型歩行訓練器，2003